SELF-DISENTANGLING DATA STORAGE TECHNIQUE

Field of the Invention:

The present invention relates to data storage. More particularly, the present invention relates to retrieval of data from a storage medium for which the data format is not necessarily known.

Background of the Invention:

Machine-readable data is generally stored digitally, as a series of logic "ones" and "zeros." For example, such data may be stored using optical, magnetic and/or electronic storage devices which may include, for example, solid-state memory devices, magnetic tape drives and optical or magnetic disk drives and arrays.

Often, it is desired to store a collection of data for later retrieval in case the original version is changed, lost or damaged. For example, a back-up copy of the data may be placed on storage media, such as a magnetic tape or floppy disk. In order for the data to be understood upon retrieval, it is generally necessary to know the format in which the data is arranged. Otherwise, the data will appear to be a random series of ones and zeros without meaning. Thus, standardized formats, such as "CPIO" (CoPy In/Out) and "TAR" (Tape ARchive) may be used for archiving data. These schemes have a drawback in that data access functionality is limited. For example, they generally require that the entire stored data structure be reconstructed. However, under certain circumstances, it may be desired to reconstruct less than the entire data structure.

Another conventional technique for archiving data is to store the data in the same format in which the software application that was used to generate the data stores and uses the data. This approach, however, has a drawback in that such application-specific data formats tend to change over time. Thus, older data formats may not be supported by newer versions of the application software.

A conventional technique for reconstructing data is to append a software routine to the beginning of the original data that reconstructs the entire original data when run over the data. For example, a simple "uncompress" program is often placed at the beginning of a piece of compressed email that is able to decode the compressed format of the email data that follows. This technique has a drawback in that it also generally requires that the entire data structure be reconstructed.

Accordingly, it would be desirable to provide a technique for the storage of machine-readable data that overcomes drawbacks associated with conventional storage and reconstruction techniques. It is to these ends that the present invention is directed.

Summary of the Invention:

The present invention is a self-disentangling data storage technique. Machine-readable data is stored by a data storage media, such as a magnetic tape or floppy drive. A software program for interpreting the data and requests for access to the data is also stored on the data storage media. The program allows the data to be retrieved using multiple different request types and interpreted in accordance with multiple different data formats. The invention overcomes disadvantages of prior storage techniques since the data can be completely or partially reconstructed, as needed. Further, the invention isolates the data storage format from the application used to generate the data so as to minimize problems caused by outdated data storage formats.

In accordance with an aspect of the invention, a method is provided for retrieving data from a data storage media. A program is loaded from the data storage media into a computer system. The program includes at least a first routine for responding to a first request type for access to the data storage media and a second routine for responding to a second request type for access to the data storage media. A request for access to data stored on the data storage media is received and a determination is made as to whether the request is of the first type or the second type. When the request is of the first type, the

first routine for accessing the data is called and, when the request is of the second type, the second routine for accessing the data is called. The requested data is then presented. The program may include information about the data, such as a file system directory. The data may be stored on the data storage media as raw data blocks.

The first routine may implement a first set of operations (e.g., including file system operations) while the second routine may implement a second set of operations (e.g., including standardized archival operations such as operations selected from CPIO and TAR). The first request type may include a request for one or more files from a file system. In which case, all or some of the data may be reformatted as a file structure. The second request type may include a request for one or more logical volumes or an image copy of the data. Further, the first request type may be by a first target system type while the second request type may be by a second target system type. In which case, presenting the requested data may include formatting the data in accordance with the target system type.

In accordance with another aspect of the invention, an article of manufacture includes a computer usable medium having data stored thereon and having computer readable program code stored thereon. The computer readable program code includes a first routine for accessing the data in response to a request for access to the data as one or more raw data blocks and a second routine for accessing the data in response to a request for access to the data as a file structure.

In accordance with yet another aspect of the invention, an article of manufacture includes a computer usable medium having data stored thereon and having computer readable program code stored thereon. The computer readable program code includes a first routine for accessing the data in response to a request from a first target system type and a second routine for accessing the data in response to a request from a second target system type.

Brief Description of the Drawings:

Figure 1 illustrates a storage media having data and a disentangling program stored thereon in accordance with the present invention;

Figure 2 illustrates a flow diagram for interpreting the data of Figure 1 in accordance with the present invention;

Figure 3 illustrates the storage media of Figure 1 in which the stored disentangling program is loaded into a computer system for interpreting the data stored on the storage media;

Figure 4 illustrates a block schematic diagram of a general-purpose computer system which may be used to interpret the data stored on the storage media; and

Figure 5 illustrates the storage media of Figures 1 and 2 for which different request types are used for accessing data stored on the media.

Detailed Description of a Preferred Embodiment:

Figure 1 illustrates a storage media 100 having data 102-106 and a disentangling program 108 stored thereon in accordance with the present invention. The media 100 may be a magnetic tape, for example. Alternately, the media be of another type such as magnetic disk (e.g., floppy or hard disk) or may use some other data storage technology, such as optical (e.g. CD or DVD) or solid-state techniques (e.g., RAM or DRAM).

The data 102-106 may be stored on the media 100 as an image back-up collection, that is, as raw data blocks that are not specially reformatted for storage. The stored software program 108 may be stored on the media 100 along with the data 102-106 and is preferably located at the beginning of the media 100, preceding the data 102-106. In addition, one or more headers may precede the stored program 108 in order to aid a reader in recognizing the program 108. Alternately, the program 108 may be stored on secondary storage associated with the media 100, such as a "smart chip" built into a magnetic tape cartridge.

The program 108 may include one or more software routines that may be invoked in response to requests for access to the data 102-106 and for interpreting and reformatting the data 102-106. The program 108 may have been compiled prior to storage on the media 100. In which case, it may be loaded and executed without further compilation. Alternately, the program may be 108 be un-compiled or only partially compiled. In which case, it may be compiled after loading, but prior to execution.

The program 108 may be written in a conventional computer language, such as C, LISP or Java. If partially pre-compiled, the program 108 may include, for example, Java byte codes. If fully compiled, the program 108 may be in the form of "X86" executable code. Languages, such as C, LISP and Java, that have well-publicized and stable virtual machine execution environments, are preferred for the program 108 over others.

The program 108 preferably provides at least two different operations for accessing or presenting the data 102-106 stored on the medium 100. For example, the program 108 may include one or more software routines or application program interfaces (APIs) to accommodate different target systems or different operation sets for accessing the data. Examples of differences in target systems that may be accommodated by the program 108 include different virtual machine architectures, different instruction sets, different computer languages, and different operating system variants. Examples of different operation sets may include standardized operations for archival, such as CPIO and TAR operations, and file system operations. Standardized archival operations include reconstruction of the entire data collection. File system operations include retrieval or generation of a file directory and retrieval of selected directory files.

In addition, the program 108 may include information about the data 102-106 stored on the media 100. This may include, for example, the amount of data 102-106, a description of its contents or a directory of files included in the data 102-106.

Further, the program 108 may perform reformatting or translation of the data 102-106 to present it to an outside entity in another format and may be used to store additional information on the media in the same format as the existing data 102-106. For example, the data 102-106 may be stored as raw data blocks or in accordance with a standard archival format, such as CPIO or TAR. The program 108 may permit access to the data 102-106 using a conventional CPIO or TAR operation and may also present the data 102-106 reformatted as a file system.

Figure 2 illustrates an exemplary flow diagram 200 for interpreting the data 102-106 of Figure 1 using the program 108 of Figure in accordance with the present invention. Program flow begins in a start state 202. From the state 202, program flow moves to a state 204 in which the program 108 may be retrieved from the storage media 100 and loaded into a computer system for interpreting the data stored on the storage media 100. This is shown in Figure 3 where the program 108 is loaded into a computer system 400.

Figure 4 illustrates a block schematic diagram of a general-purpose computer system 400 by which the present invention may be implemented. The computer system 400 may include a general-purpose processor 402, a memory 404, such as persistent memory (e.g., a hard disk for program memory) and transitory memory (e.g., RAM for storing the device tree 100), a communication bus 406, and input/output devices 408, such as a keyboard, monitor and mouse. The memory 404 may include memory devices such as a magnetic tape reader for reading the storage media 100 of Figure 1. The computer system 400 is conventional. As such, it will be apparent that the system 400 may include more or fewer elements than shown in Figure 4 and that other elements may be substituted for those illustrated in Figure 4.

Referring again to Figure 2, program flow moves from the state 204 to a state 206. In the state 206, the computer 400 may compile the program 108 and/or pass the program 108 to an interpreter, as necessary. For this purpose, a compiler or interpreter program may also be stored in the memory 404 of the computer system 400.

From the state 206, program flow moves to a state 208 in which a handle may be passed from the media 100 to the computer 400 for use by the program 108 for accessing the data 102-106. This handle may be a tag or reference to the data 102-106 rather than the data 102-106, itself.

Then, in a state 210, a determination may be made as to whether a request has been received for access to the data 102-106. For example, an application program operating on the computer system 400 may require access to some or all of the data 102-106. Alternately, a user may initiate access via the computer 400. Program flow may remain in the state 210 until such a request is received. When a request for access to the data 102-106 is received, program flow moves to a state 212. In the state 212, a determination may be made as to the type of request which was received in the state 210.

The request received in the state 210 may be, for example, one of two principal types. One type of request may be to access the data 102-106 as though the data 102-106 were an image backup or a set of logical volumes. Accordingly, this type of request may assume that the data 102-106 is stored according to a standardized archival format, such as CPIO or TAR. An example of this type of request is a request to reconstruct the entire data collection to another location, such as onto a hard-drive of the computer system 400 (Figure 4). Another example of this type of request may be to access specified logical blocks (e.g., "the first 50 blocks of logical volume 23") of the data 102-106.

Assuming the request received in the state 210 is to access the data 102-106 as an image back-up or logical volume, then program flow moves from the state 212 to a state 214, in which an application program interface (API) routine may be called that is appropriate to this type of request.

Another principal type of request may be to access the data 102-106 as though the data 102-106 were a file system. An example of this type of request is a request to retrieve a specified file (e.g., by file name: "XXX/YYY"). Another example of this type of request is a request to list the contents of the media 100 as a set of files (i.e. a file directory). A further example of this type

of request may be to reconstruct the entire data collection 102-106 as a file system (e.g., to be stored on the hard drive of the computer system 400). Assuming the data 102-106 is stored as raw data blocks, this type of request may require reformatting of the data 102-106 to a file system format.

Assuming the request received in the state 210 is to access the data 102-106 as a file system, then program flow moves from the state 212 to a state 216, in which an application program interface (API) routine may be called that is appropriate to this type of request.

From either state 214 or 216, program flow moves to a state 218 in which the requested data may be returned to the requesting application. From the state 218, program flow returns to the state 210 where it awaits another request for access. While in the state 210, one or more additional requests to access the data may be received. In response, program flow returns to the program loop including states 212, 214, 216 and 218. Note that the program 108 (Figure 1) can accommodate different types of requests. Accordingly, the successive requests received in the state 210 may be any combinations of requests that the program 108 can carry-out. Thus, one request may be for a specific file (a file-system type request), while the next successive request may be for a complete data reconstruction (an image backup request). Thus, as shown in Figure 5, different request types (e.g., "request1" and "request2") may be received by the computer system 400 for accessing data stored on the media 100.

A specific example of the invention is illustrative. Assume it is desired to create an archival copy of a data collection. Rather than merely write the raw data blocks onto the archival storage media, a software program (e.g., the program 108 written in Java) is also stored on the storage media. This software program may allow the data to be read as both (a) a set of raw data blocks; and (b) a UNIX file system. The API for a set of operations for reading raw data blocks may provide for conventional CPIO or TAR operations. The API for a set of file system operations may be that of a UNIX virtual file system (VFS) interface, which allows the storage media 100 to be used as virtual file system

when the program is loaded into memory 404. This would allow future readers of the media 100 to understand its content, even if, for example, the directory format of operating system had changed. By using the VFS interface, it would be possible for a single file to be recovered from an image-based back-up copy (i.e. a raw block copy) without first having to download the entire contents to a hard-disk. For increased performance, additional information, such as a directory map, may be stored on the media 100 prior to the raw data blocks 102-106. This directory map could be hidden during requests using the raw data block API.

Other requests may be received in the state 210. For example, the program may include routines for accommodating different target systems. In which case, requests for access to the data 102-106 that are generated by applications running on the computer 400 may be received in various different forms, depending upon the instruction sets, virtual machine architectures, languages or operating systems that may be run on the computer 400. In response, the retrieved data 102-106 may be presented differently to different target systems, even if essentially the same request is received. The target system type may be identified based on the form of the request. Thus, for example, a request for a specific file stored on the media 100 may return the specified file, but formatted differently depending on the type of target system that made the request.

Once all of the requests to access the data are completed, program flow may move from the state 210 to a state 220. In the state 220, the program 108 (Figure 1) may be closed. Then, program flow may terminate in a state 222.

While the foregoing has been with reference to particular embodiments of the invention, it will be appreciated by those skilled in the art that changes in these embodiments may be made without departing from the principles and spirit of the invention, the scope of which is defined by the appended claims.